Wavelets on Streams

Minos Garofalakis
Technical University of Crete
minos@acm.org

SYNONYMS

None.

DEFINITION

Unlike conventional database query-processing engines that require several passes over a static data image, streaming data-analysis algorithms must often rely on building concise, approximate (but highly accurate) *synopses* of the input stream(s) in real-time (i.e., in one pass over the streaming data). Such synopses typically require space that is significantly sublinear in the size of the data and can be used to provide *approximate query answers*.

The collection of the top (i.e., largest) coefficients in the *wavelet transform* (or, *decomposition*) of an input data vector is one example of such a key feature of the stream. *Wavelets* provide a mathematical tool for the hierarchical decomposition of functions, with a long history of successful applications in signal and image processing [10]. Applying the wavelet transform to a (one- or multi-dimensional) data vector and retaining a select small collection of the largest wavelet coefficient gives a very effective form of lossy data compression. Such *wavelet summaries* provide concise, general-purpose summaries of relational data, and can form the foundation for fast and accurate approximate query processing algorithms.

HISTORICAL BACKGROUND

Haar wavelets have recently emerged as an effective, general-purpose data-reduction technique for approximating an underlying data distribution, and providing a foundation for approximate query processing over traditional (static) relational data. Briefly, the idea is to apply the Haar wavelet decomposition to the input relation to obtain a compact summary that comprises a select small collection of *wavelet coefficients*. The results of [3, 9, 11] have demonstrated that fast and accurate approximate query processing engines can be designed to operate solely over such compact wavelet summaries. Wavelet summaries can also give accurate *histograms* of the underlying data distribution at multiple levels of resolution, thus providing valuable primitives for effective data visualization.

In the setting of static relational tables, the Haar wavelet transform is well understood, and scalable wavelet decomposition algorithms for constructing wavelet summaries have been known for some time [3, 11]. Typically, such algorithms assume at least linear space and several passes over the data. Data streaming models introduce the novel challenge of maintaining the topmost wavelet coefficients over a dynamic data distribution (rendered as a stream of updates), while only utilizing *small space and time* (i.e., significantly sublinear in the size of the data distribution),

SCIENTIFIC FUNDAMENTALS

The Haar Wavelet Decomposition. Consider the one-dimensional data vector $a = [2, 2, 0, 2, 3, 5, 4, 4]$ comprising $N = 8$ data values. In general, N denotes the data set size and $[N]$ is defined as the integer index domain $\{0, \ldots, N-1\}$; also, without loss of generality, N is assumed to be a power of 2 (to simplify notation). The Haar Wavelet Transform (HWT) of a is computed as follows. The data values are first averaged together pairwise to get a new "lower-resolution" representation of the data with the pairwise averages $[\frac{2+2}{2}, \frac{0+2}{2}, \frac{3+5}{2}, \frac{4+4}{2}] = [2, 1, 4, 4]$. This averaging loses some of the information in a. To restore the original a values, *detail coefficients* that capture the missing information are needed. In the HWT, these detail coefficients are the differences of the (second of the) averaged values from the computed pairwise average. Thus, in this simple example, for the first pair of averaged values, the detail coefficient is 0 since $\frac{2-2}{2} = 0$, for the second it is -1 since $\frac{0-2}{2} = -1$. No information is lost in this process – one can reconstruct the eight values of the original data array from the lower-resolution array containing the four averages and the four detail coefficients. This pairwise averaging and differencing process is recursively applied on the lower-resolution array of averages until the overall average is reached, to get the full Haar decomposition. The final HWT of a is given by $w_a = [11/4, -5/4, 1/2, 0, 0, -1, -1, 0]$, that is, the overall average followed by the detail coefficients in order of increasing resolution. Each entry in w_a is called a *wavelet coefficient*. The main advantage of using w_a instead of the original data vector a is that for vectors containing similar values most of the detail coefficients tend to have very small values. Thus, eliminating such small coefficients from the wavelet transform (i.e., treating them as zeros) introduces only small errors when reconstructing the original data, resulting in a very effective form of lossy data compression [10].

A useful conceptual tool for visualizing and understanding the HWT process is the *error tree* structure [9] (shown in Fig. 1 for the example array a). Each internal tree node c_i corresponds to a wavelet coefficient (with the root node c_0 being the overall average), and leaf nodes $a[i]$ correspond to the original data-array entries. This view allows us to see that the reconstruction of any $a[i]$ depends only on the $\log N + 1$ coefficients in the path between the root and $a[i]$; symmetrically, it means a change in $a[i]$ only impacts its $\log N + 1$ ancestors in an easily computable way. The *support* for a coefficient c_i is defined as the contiguous range of data-array that c_i is used to reconstruct (i.e., the range of data/leaf nodes in the subtree rooted at c_i). Note that the supports of all coefficients at resolution level l of the HWT are exactly the 2^l (disjoint) *dyadic ranges* of size $N/2^l = 2^{\log N - l}$ over $[N]$, defined as $R_{l,k} = [k \cdot 2^{\log N - l}, \ldots, (k+1) \cdot 2^{\log N - l} - 1]$ for $k = 0, \ldots, 2^l - 1$ (for each resolution level $l = 0, \ldots, \log N$). The HWT can also be conceptualized in terms of vector inner-product computations: let $\phi_{l,k}$ denote the vector with $\phi_{l,k}[i] = 2^{l - \log N}$ for $i \in R_{l,k}$ and 0 otherwise, for $l = 0, \ldots, \log N$ and $k = 0, \ldots, 2^l - 1$; then, each of the coefficients in the HWT of a can be expressed as the inner product of a with one of the N distinct Haar *wavelet basis vectors*:

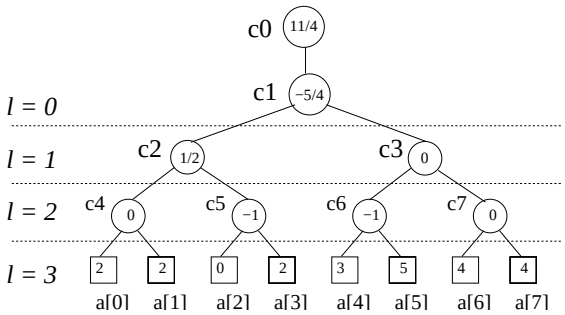

Figure 1: Error-tree structure for the example data array a ($N = 8$).

$$\{\tfrac{1}{2}(\phi_{l+1,2k} - \phi_{l+1,2k+1}) : l = 0, \ldots, \log N - 1; k = 0, \ldots, 2^l - 1\} \cup \{\phi_{0,0}\}$$

Intuitively, wavelet coefficients with larger support carry a higher weight in the reconstruction of the original data values. To equalize the importance of all HWT coefficients, a common normalization scheme is to scale the coefficient values at level l (or, equivalently, the basis vectors $\phi_{l,k}$) by a factor of $\sqrt{N/2^l}$. This normalization essentially turns the HWT basis vectors into an *orthonormal basis* — letting c_i^* denote the normalized coefficient values, this fact has two important consequences: (1) The *energy* (squared L_2-norm) of the a vector is preserved in the wavelet domain, that is, $||a||_2^2 = \langle a, a \rangle = \sum_i a[i]^2 = \sum_i (c_i^*)^2$ (by Parseval's theorem); and, (2) Retaining the B largest coefficients in *absolute normalized value* gives the (provably) best B-term approximation in terms of L_2 (or, sum-squared) error in the data reconstruction (for a given budget of coefficients B) [10].

The HWT and its key properties also naturally extend to the case of *multi-dimensional* data distributions; in that case, the input a is a d-dimensional data array, comprising N^d entries[1], and the HWT of a results in a d-dimensional wavelet-coefficient array w_a with N^d coefficient entries. The supports of d-dimensional Haar coefficients are d-dimensional hyper-rectangles (over dyadic ranges in $[N]^d$), and can be naturally arranged in a generalized error-tree structure [5, 6].

Wavelet Summaries on Streaming Data. Abstractly, the goal is to continuously track a compact summary of the B topmost wavelet coefficient values for a dynamic data distribution vector a rendered as a continuous stream of updates. Algorithms for this problem should satisfy the key *small space/time* requirements for streaming algorithms; more formally, streaming wavelet algorithms should (ideally) guarantee (1) *sublinear space usage* (for storing a synopsis of the stream), (2) *sublinear per-item update time* (to maintain the synopsis), and (3) *sublinear query time* (to produce a, possibly approximate, wavelet summary), where "sublinear" typically means polylogarithmic in the domain size N. The streaming wavelet summary construction problem has been examined under two distinct data streaming models.

• *Wavelets in the Ordered Aggregate (Time Series) Streaming Model.* Here, the entries of the input data vector a are rendered over time in the increasing (or, decreasing) order of the index domain values. This means, for instance, that $a[1]$ (or, the set of all updates to $a[1]$) is seen first, followed by $a[2]$, then $a[3]$, and so on. In this case, the set of the B topmost HWT values over a can be maintained *exactly* in small space and time, using a simple algorithm based on the error-tree structure (Fig. 1) [7]: Consider reading (an update to) item $a[i+1]$ in the stream; that is, all items $a[j]$ for $j \leq i$ have already streamed through. The algorithm maintains the following two sets of (partial) coefficients:

1. Highest B HWT coefficient values for the portion of the data vector seen thus far; and,
2. $\log N + 1$ *straddling* partial HWT coefficients, one for each level of the error tree. At level l, index i *straddles* the HWT basis vector $\phi_{l,k}$, where $j \in [k \cdot 2^{\log N - l}, (k+1) \cdot 2^{\log N - l} - 1]$. (Note that there is *at most one* such basis vector per level.)

When the $(i+1)^{th}$ data item is read, the value for each of the affected straddling coefficients is updated. With the arrival of the $(i+1)^{th}$ item, some coefficients may no longer be straddling (i.e., their computation is now complete). In that case, the value of these coefficients is compared against the the current set of B highest coefficients, and only the B largest coefficient values in the combined set are retained. Also, for levels where a straddling coefficient has been completed, a new straddling

[1]Without loss of generality, a domain of $[N]^d$ is assumed for the d-dimensional case.

coefficient is initiated (with an initial value of 0). In this manner, at every position in the time series stream, the set of the B topmost HWT coefficients is retained exactly.

Theorem 1 ([7]) *The topmost B HWT coefficients can be maintained exactly in the ordered aggregate (time series) streaming model using $O(B + \log N)$ space and $O(B + \log N)$ processing time per item.*

Similar algorithmic ideas can also be applied to more sophisticated wavelet thresholding schemes (e.g., that optimize for error metrics other than L_2) to obtain space/time efficient wavelet summarization algorithms in the ordered aggregate model [8].

• *Wavelets in the General Turnstile Streaming Model.* Here, updates to the data vector a can appear in any arbitrary order in the stream, and the final vector entries are obtained by implicitly aggregating the updates for a particular index domain value. More formally, in the turnstile model, each streaming update is a pair of the form $(i, \pm v)$, denoting a net change of $\pm v$ in the $a[i]$ entry; that is, the effect of the update is to set $a[i] \leftarrow a[i] \pm v$. (The model naturally generalizes to multi-dimensional data: for d data dimensions, each update $((i_1, \ldots, i_d), \pm v)$ effects a net change of $\pm v$ on entry $a[i_1, \ldots, i_d]$.) The problem of maintaining an accurate wavelet summary becomes significantly more complex when moving to this much more general streaming model. Gilbert et al. [7] prove a strong *lower bound* on the space requirements of the problem: for arbitrary turnstile streaming vectors, *nearly all* of the data must be stored to recover the top B HWT coefficients.

Existing solutions for wavelet maintenance over turnstile data streams rely on randomized schemes that return only an *approximate* synopsis comprising (at most) B Haar coefficients that is provably near-optimal (in terms of the captured energy of the underlying vector) assuming that the data vector satisfies the *"small-B property"* (i.e., most of its energy is concentrated in a small number of HWT coefficients) — this assumption is typically satisfied for most real-life data distributions [7]. One of the key ideas is to maintain a randomized *AMS sketch* [2], a broadly applicable stream synopsis structure comprising randomized linear projections of the streaming data vector a. Briefly, an *atomic AMS sketch* of a is simply the *inner product* $\langle a, \xi \rangle = \sum_i a[i]\xi(i)$, where ξ denotes a random vector of four-wise independent ± 1-valued random variates.

Theorem 2 ([1, 2]) *Consider two (possibly streaming) data vectors a and b, and let Z denote the $O(\log(1/\delta))$-wise median of $O(1/\epsilon^2)$-wise means of independent copies of the atomic AMS sketch product $(\sum_i a[i]\xi_j(i))(\sum_i b[i]\xi_j(i))$. Then, $|Z - \langle a, b\rangle| \leq \epsilon||a||_2||b||_2$ with probability $\geq 1 - \delta$.*

Thus, using AMS sketches comprising only $O(\frac{\log(1/\delta)}{\epsilon^2})$ atomic counters, the vector inner product $\langle a, b\rangle$ can be approximated to within $\pm\epsilon||a||_2||b||_2$ (hence implying an ϵ-relative error estimate for the squared L_2 norm $||a||_2^2$).

Since Haar coefficients of a are inner products with a fixed set of wavelet-basis vectors, the above theorem forms the key to developing efficient, approximate wavelet maintenance algorithms in the turnstile model. Gilbert et al. [7] propose a solution (termed "GKMS" in the remainder of the discussion) that focuses primarily on the one-dimensional case. GKMS maintains an AMS sketch for the streaming data vector a. To produce the approximate B-term representation, GKMS employs the constructed sketch of a to estimate the inner product of a with *all wavelet basis vectors*, essentially performing an exhaustive search over the space of all wavelet coefficients to identify important ones. More formally, assuming that there is a B-coefficient approximate representation of the signal with energy at least $\eta||a||_2^2$ ("small B property"), the GKMS algorithm uses a maintained AMS sketch to exhaustively estimate each Haar coefficient and selects up to B of the largest coefficients (excluding those whose square is less than $\eta\epsilon||a||_2^2/B$, where $\epsilon < 1$ is the desired accuracy guarantee). GKMS also uses techniques based on range-summable random variables constructed using Reed-Muller codes to reduce or amortize the cost of this exhaustive search by allowing the sketches of basis vectors (with potentially large supports) to be computed more quickly.

Theorem 3 ([7]) *Assuming there exists a B-term representation with energy at least $\eta||a||_2^2$, then, with probability at least $1 - \delta$, the GKMS algorithm finds a representation of at most B coefficients that captures at least $(1 - \epsilon)\eta$ of the signal energy $||a||_2^2$, using $O(\log^2 N \log(N/\delta)B^2/(\eta\epsilon)^2)$ space and per-item processing time.*

A potential problem lies in the query time requirements of the GKMS algorithm: even with the Reed-Muller code optimizations, the overall query time for discovering the top coefficients remains superlinear in N (i.e., at least $\Omega(\frac{1}{\epsilon^2}N \log N)$), violating the third requirement on streaming schemes. This also renders direct extensions of GKMS to multiple dimensions infeasible since it implies an exponential explosion in query cost (requiring at least $O(N^d)$ time to cycle through all coefficients in d dimensions). In addition, the update cost of the GKMS algorithm is *linear in the size of the sketch* since the whole data structure must be "touched" for each update. This is problematic for high-speed data streams and/or even moderate sized sketch synopses.

To address these issues, Cormode et al. [5] propose a novel solution that relies on two key technical ideas. First, they work *entirely in the wavelet domain*: instead of sketching the original data entries, their algorithms sketch the wavelet-coefficient vector w_a as updates arrive. This avoids any need for complex range-summable hash functions (i.e., Reed-Muller codes).

Second, they employ *hash-based grouping* in conjunction with *efficient binary-search-like techniques* to enable very fast updates as well as identification of important coefficients in polylogarithmic time.

– *Sketching in the Wavelet Domain.* The first technical idea in [5] relies on the observation that it is possible efficiently produce sketch synopses of the stream *directly in the wavelet domain*. That is, the impact of each streaming update can be translated on the relevant wavelet coefficients. By the linearity properties of the HWT and the earlier description, an update to the data entries corresponds to only polylogarithmically many coefficients in the wavelet domain. Thus, on receiving an update to a, it can be directly converted to $O(\text{polylog}(N))$ updates to the wavelet coefficients, and an approximate (sketch) representation of the wavelet coefficient vector w_a can be maintained.

– *Time-Efficient Updates and Large-Coefficient Searches.* Sketching in the wavelet domain means that, at query time, an approximate representation of the wavelet-coefficient vector w_a is available, and can be employed to identify all those coefficients that are "large", relative to the total energy of the data $\|w_a\|_2^2 = \|a\|_2^2$. While AMS sketches can provide such estimates (a point query is just a special case of an inner product), querying remains much too slow taking at least $\Omega(\frac{1}{\epsilon^2}N)$ time to find which of the N coefficients are the B largest. Instead, the schemes in [5] rely on a *divide-and-conquer* or *binary-search-like* approach for finding the large coefficients. This requires the ability to efficiently estimate sums-of-squares for *groups* of coefficients, corresponding to dyadic subranges of the domain $[N]$. Low-energy regions can then be disregarded, recursing only on high-energy groups — this guarantees no false negatives, as a group that contains a high-energy coefficient will also have high energy as a whole. The algorithms of [5] also employ *randomized, hash-based grouping* of dyadic groups and coefficients to guarantee that each update only touches a small portion of the synopsis, thus guaranteeing very fast update times.

The key to the Cormode et al. solution is a hash-based probabilistic synopsis data structure, termed *Group-Count Sketch (GCS)*, that can estimate the energy of fixed groups of elements from a vector w of size N under the turnstile streaming model [5]. This translates to several streaming L_2-norm estimation problems (one per group). A simple solution would be to keep an AMS sketch of each group separately; however, there can be *many* (e.g., linear in N) groups, implying space requirements that are $O(N)$. Streaming updates should also be processed as quickly as possible. The GCS synopsis requires small, sublinear space and takes sublinear time to process each stream update item; more importantly, a GCS can provide a high-probability estimate of the energy of a group within additive error $\epsilon\|w\|_2^2$ in *sublinear time*. In a nutshell, the GCS synopsis first partitions items of w into their group using an `id()` function (which, in the case of Haar coefficients, is trivial since it corresponds to fixed dyadic ranges over $[N]$), and then randomly maps groups to buckets using a hash function $h()$. Within each bucket, a second stage of hashing of items to sub-buckets is applied (using another hash function $f()$, where each contains an atomic AMS sketch counter in order to estimate the L_2 norm of the elements in the bucket. As with most randomized estimation schemes, a GCS synopsis comprises t independent instantiations of this basic randomized structure, each with independently chosen hash function pairs $(h(), f())$ and ξ families for the AMS estimator; during maintenance, a streaming update (x, u) is used to update each of the t AMS counters corresponding to element x. (A pictorial representation is shown in Fig. 2.) To estimate the energy of a group g, for each independent instantiation $m = 1, \ldots, t$ of the bucketing structure, the squared values of all the AMS counters in the sub-buckets corresponding to bucket $h_m(g)$ are summed, and then the *median* of these t values is returned as the estimate.

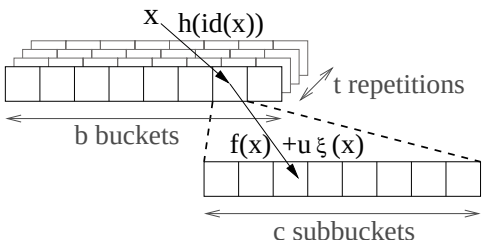

Figure 2: The GCS data structure: Element x is hashed (t times) to a bucket of groups (using $h(\texttt{id}(x))$) and then a sub-bucket within the bucket (using $f(x)$), where an AMS counter is updated.

Theorem 4 ([5]) *The GCS can estimate the energy of item groups of the vector w within additive error $\epsilon\|w\|_2^2$ with probability $\geq 1-\delta$ using space of $O\left(\frac{1}{\epsilon^3}\log\frac{1}{\delta}\right)$ counters, per-item update time of $O\left(\log\frac{1}{\delta}\right)$, and query time of $O\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$.*

To recover coefficients with large energy in the w vector, the algorithm employs a *hierarchical search-tree structure* on top of $[N]$: Each level in this tree structure induces a certain partitioning of elements into groups (corresponding to the nodes at that level), and per-level GCS synopses can be used to efficiently recover the high-energy groups at each level (and, thus, quickly zero in on high-energy Haar coefficients). Using these ideas, Cormode et al. [5] demonstrate that the accuracy guarantees of Theorem 3 can be obtained using $O(\frac{B^3\log N}{\epsilon^3\eta^3}\cdot\log\frac{B\log N}{\epsilon\eta\delta})$ space, $O(\log^2 N\cdot\log\frac{B\log N}{\epsilon\eta\delta})$ per item processing time, *and* $O(\frac{B^3}{\epsilon^3\eta^3}\cdot\log N\cdot\log\frac{B\log N}{\epsilon\eta\delta})$ query time. In other words, their GCS-based solution guarantees sublinear space *and* query time, as well as per-item processing times that are sublinear *in the size of the stream synopsis*. Their results also naturally extend to the multi-dimensional case [5].

KEY APPLICATIONS

Wavelet-based summaries are a general-purpose data-reduction tool, and the maintenance of such summaries over continuous

data streams has several important applications, including large-scale IP network monitoring and network-event tracking (e.g., for detecting network traffic anomalies or Denial-of-Service attacks), approximate query processing over warehouse update streams, clickstream and transaction-log monitoring in large web-server farms, and satelite- or sensornet-based environmental monitoring.

DATA SETS

Several publicly-available real-life data collections have been used in the experimental study of streaming wavelet summaries; examples include the US Census Bureau data sets (`http://www.census.gov/`), the UCI KDD Archive (`http://kdd.ics.uci.edu/`), and the UW Earth Climate and Weather Data Archive (`http://www-k12.atmos.washington.edu/k12/grayskies/`).

FUTURE DIRECTIONS

The area of streaming wavelet-based summaries is rich with interesting algorithmic questions. The bulk of the discussion here focuses on L_2-error synopses. The problem of designing efficient streaming methods for maintaining wavelet summaries that optimize for *non-L_2 error metrics* (e.g., general L_p error) under a turnstile streaming model remains open. Optimizing for non-L_2 error implies more sophisticated coefficient thresholding schemes based on dynamic programming over the error-tree structure (e.g., [6, 8]); while such methods can be efficiently implemented over ordered aggregate streams [8], no space/time efficient solutions are known for turnstile streams. Dealing with *physically-distributed streams* also raises interesting issues for wavelet maintenance, such as trading off approximation quality with *communication* (in addition to time/space). The distributed AMS sketching algorithms of Cormode and Garofalakis [4] can be applied to maintain an approximate wavelet representation over distributed turnstile streams, but several issues (e.g., appropriate prediction models for local sites) remain open. Finally, from a systems perspective, the problem of incorporating wavelet summaries in industrial-strength data-streaming engines, and testing their viability in real-life scenarios remains open.

CROSS REFERENCES

WAVELETS IN DATABASE SYSTEMS, APPROXIMATE QUERY PROCESSING, DATA COMPRESSION, DATA REDUCTION, SYNOPSIS STRUCTURES, DATA SKETCH/SYNOPSIS

References

[1] Noga Alon, Phillip B. Gibbons, Yossi Matias, and Mario Szegedy. "Tracking Join and Self-Join Sizes in Limited Storage". In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Philadeplphia, Pennsylvania, May 1999.

[2] Noga Alon, Yossi Matias, and Mario Szegedy. "The Space Complexity of Approximating the Frequency Moments". In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, Philadelphia, Pennsylvania, May 1996.

[3] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. "Approximate Query Processing Using Wavelets". In *Proceedings of the 26th International Conference on Very Large Data Bases*, Cairo, Egypt, September 2000.

[4] Graham Cormode and Minos Garofalakis. "Approximate Continuous Querying over Distributed Streams". *"ACM Transactions on Database Systems"*, 30(4), December 2005.

[5] Graham Cormode, Minos Garofalakis, and Dimitris Sacharidis. "Fast Approximate Wavelet Tracking on Streams". In *Proceedings of the 10th International Conference on Extending Database Technology (EDBT'2006)*, Munich, Germany, March 2006.

[6] Minos Garofalakis and Amit Kumar. "Wavelet Synopses for General Error Metrics". *ACM Transactions on Database Systems*, 30(4), December 2005.

[7] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin J. Strauss. "One-pass wavelet decomposition of data streams". *IEEE Transactions on Knowledge and Data Engineering*, 15(3):541–554, May 2003.

[8] Sudipto Guha and Boulos Harb. "Wavelet synopsis for data streams: minimizing non-euclidean error". In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Chicago, Illinois, August 2005.

[9] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. "Wavelet-Based Histograms for Selectivity Estimation". In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, June 1998.

[10] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *"Wavelets for Computer Graphics – Theory and Applications"*. Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[11] Jeffrey Scott Vitter and Min Wang. "Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets". In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, Philadelphia, Pennsylvania, May 1999.